POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

HobbyBuddy

Author(s): **Marta Radaelli - 10657046**
Francesco Scandale - 10616610

Academic Year: 2022-2023

Contents

1 Introduction

1.1. HobbyBuddy

HobbyBuddy is a multi-device application for smartphone and tablet that provides users with an easy-to-use platform to discover new hobbies and mentors.
HobbyBuddy can be adopted for both personal growth and social purposes as it can be used to discover a new passion and to learn new skills by following mentors' tutorials or by meeting them and fellow aficionados in in-person sessions. At the same time, the platform can also be used to keep track of one's improvement, while the user can also make new friends and be inspired by their favourite hobbies and latest improvements.

This document explains the most important design choices we made while developing HobbyBuddy and the motivations behind them, while also showcasing the intensive and successful testing campaign executed on the app.

1.2. Requirements

The following list contains the requirements that the application should satisfy.

ID	Functional Requirements
FR1	Users should be able to sign in/sign up in the app
FR2	Users should be able to logout from the app
FR3	Registered users should be able to add and remove hobby subscription
FR4	Registered users should be able to add and remove favourite mentors
FR5	Registered users should be able to check the list of upcoming classes of mentors
FR6	Registered users should be able to check the list of available tutorials of the mentors
FR7	Registered users should be able to view the tutorials of the mentors
FR8	Registered users should be able to check the location of liked mentors' classes on the map
FR9	Registered users should be able to check the details of liked mentors' classes on the map
FR10	Registered users should be able to check the list of liked hobbies
FR11	Registered users should be able to check the list of liked mentors
FR12	Registered users should be able to upload milestones

FR13	Registered users should be able to check the uploaded milestones
FR14	Registered users should be able to edit their personal information
FR15	Registered users should be able to see their list of friends
FR16	Registered users should be able to check a friend's profile
FR17	Registered users should be able to send a friendship request to other users
FR18	Registered users should be able to accept or decline a friendship request from other users
FR19	Registered users should be able to remove a friend from their list

1.3. Non-functional Requirements

Some non-functional requirements must be met in order for **HobbyBuddy** to perform as expected:

- The application must be easy to use and intuitive.
- The application must be fast and responsive.
- The application must be available 24/7 with minimal maintenance downtime.
- The application must be secure and protect user data..

1.4. Implemented Features

The functionalities implemented in the app, trying to follow as much as possible a likely order of interaction, are:

1.4.1. Log in and sing up

Users can sign in (or sign up) by using a unique username, a unique email and a password.

1.4.2. Search and find available hobbies on the home page

Registered users can view all the available hobbies from a list that is displayed in the home page and save them by tapping the appropriate button on the hobby home page. A like enables the user to quickly access the liked hobbies from the profile page.

1.4.3. Add/remove hobby/mentor from favorites

Registered users can view all the mentors available for a hobby from a list that is displayed in the hobby page and like them by tapping the appropriate button. A like enables the user to quickly access the liked mentors from the profile page.

1.4.4. Find mentor class on the map

Registered users can find the location and information of nearby upcoming classes of their liked mentors on the on-app map.

1.4.5. View list of friends/users and add/remove friends

Registered users can view their list of friends, search for one in particular, view their personal page or remove a friend. They can also view the list of all registered users, search for one in particular, view their personal page and send a friend request. The friends page has an icon button that notifies the user of any pending friendship request that was received and the user can decide whether to accept or decline the request.

1.4.6. View personal profile and upload new milestones

Registered users can view their profile page with all liked hobbies and mentors and their respective pages. In the page, the user can visualize all milestone that where uploaded and add a new milestone with an image and a caption.

1.4.7. View a mentors' profile and follow a class

Registered users can view all mentors' profile, where the like button is available as well, see the list of upcoming in-person classes and follow one of their tutorials based on the difficulty level and the users' experience.

1.4.8. Edit user settings, password and preferences

Registered users can change their password and modify their profile information in the settings page of the app. In this section it is possible to edit the background and profile pictures, as well as first and last name.

It is also possible to change the theme (light/dark) of the application.

2 | Architectural Design

2.1. General Overview

We have built HobbyBuddy using Flutter, an open source framework developed by Google for building multi-platform applications from a single codebase. Dart is the language on which Flutter is based on, optimized to create fast apps on any platform.
For the development of the application, different services, when possible, are used and integrated by exploiting asynchronous communication protocols that, with the help of future builders, allows the application to be as responsive and smooth as possible regardless of the performance of the external services.

2.1.1. Client-server paradigm

The architecture designed for the application can be framed in the client-server paradigm. The user is supposed to interact with client side of the app by means of his device, while the data accessed by the user is, on the other hand, kept on a server: the client communicates with the server through REST APIs provided, among others, by Firebase.
The system architecture of a Flutter app with Firebase as a backend can be described in three main layers: UI layer, business logic layer, and service layer.

- The **UI Layer** consists of Flutter widgets, which are the fundamental building blocks of Flutter user interfaces. Flutter allows for reactive user interfaces, meaning that the UI can automatically update as the underlying data changes.
- The **Business Logic Layer** contains all app business and presentation logic.
- The **Service Layer** includes all services needed to interact with third-party services such as APIs (for example Firebase SDK in the Firebase case).

2.1.2. Shared Preferences

Shared preferences is a plugin that provides persistent storage for simple data represented in key-value format in device's memory.
In this application it has been useful for storing users' preferences and data: when a user updates

one of these options in the dedicated page, the new value is assigned to the specific key and stored in memory so that, when the user closes and reopens the application, the chosen settings are always restored. It also used to store a user's data in order to fetch it more quickly and avoid making calls to the database.

2.2. External Services

HobbyBuddy exploits multiple external services in order to implement its functionalities.

2.2.1. Google Maps for flutter

Google Maps for Flutter is a flutter plugin for integrating Google Maps in iOS and Android applications. It provides Flutter with a GoogleMap widget that can be highly customized, based on the needs of the application.

2.2.2. Google Firebase

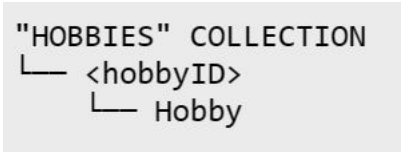
Google Firebase is the main suite of services adopted for back-end data management. Within this suite, the used services are:

- **Cloud Firestore** is a document-based database that stores all the relevant data displayed in the application. Once a user signs up, the main information about its profile are saved in Cloud Firestore.
- **Cloud Storage** is built for apps that need to store and serve users' files, such as photos or videos. In this scenario, it has been mainly used to store users' profile pictures, milestones uploads and multimedia of mentors' courses.
- **Firebase Authentication** is a back-end service provider that authenticates users to the app. It natively supports authentication using email and password, but it has been modified so that the users can login via their unique username and password. Once a user signs up, the login information is stored in Firebase Authentication.

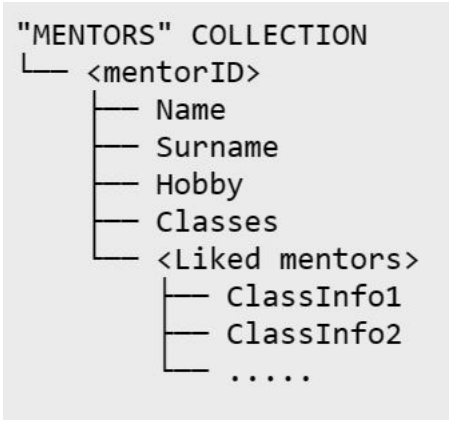
2.3. Database Design

As previously explained, HobbyBuddy uses Firestore services to store all data. The following collections where designed to implement all the application's requirements:

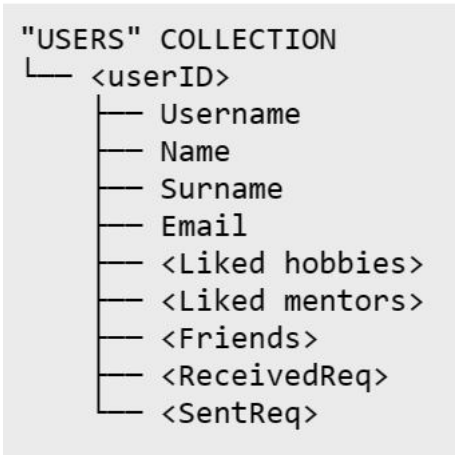
- "**Hobbies**" collection: used to keep track of all the hobbies that exist within the scope of our application



- "**Mentors**" collection: each mentor has a dedicated document id with a docID and the fields as in the sketch below;



- "**Users**" collection: each user has a dedicated document with its own unique id and each document contains the profile information shown in the picture below;



- "**Markers**" collection: defines the position of a lesson and the hobby image (defined by field "title" needed to put the marker on the maps screen;

```
"MARKERS" COLLECTION
└── <markerID>
    ├── Title
    ├── Lat
    ├── Long
    └── Mentor
```

All images, videos and textual resources are, on the other hand, stored on the firebase cloud storage in two separate folders depending on whether they are user's or mentor's materials. The structures of the folders are shown in the pictures below.

```
Users/
├── Username1/
│   ├── background.jpg
│   ├── propic.jpg
│   └── milestones/
│       ├── timestamp1/
│       │   ├── caption.txt
│       │   └── pic.jpg
│       └── timestamp2/
│           ├── caption.txt
│           └── pic.jpg
├── Username2/
│   ├── background.jpg
│   ├── propic.jpg
│   └── milestones/
│       ├── timestamp1/
│       │   ├── caption.txt
│       │   └── pic.jpg
│       └── timestamp2/
│           ├── caption.txt
│           └── pic.jpg
└── .....
```

Figure 2.1: users storage

```
Mentors/
├── Name Surname 1/
│   ├── background.jpg
│   ├── propic.jpg
│   └── courses/
│       ├── timestamp1/
│       │   ├── title.txt
│       │   ├── text.txt
│       │   ├── pic.jpg
│       │   ├── video1.mp4
│       │   ├── picture1.jpg
│       │   ├── picture2.jpg
│       │   └── ....
│       └── timestamp2/
│           ├── title.txt
│           └── ....
└── Name Surname 2/
    ├── background.jpg
    ├── propic.jpg
    └── courses/
        ├── timestamp1/
        │   ├── title.txt
        │   └── ....
        └── timestamp2/
            ├── title.txt
            └── ....
```

Figure 2.2: mentors' storage

2.4. Main Dependencies

The application also expands its functionalities thanks to public external libraries available on pub.dev:

firebase_core	Necessary to initialize Firebase services
firebase_auth	Used for authentication with email and password
cloud_firestore	Needed to use the Cloud Firestore API
firebase_storage	Used to access the Firebase Cloud Storage API
shared_preferences	Used to provide persistent storage for simple data
image_picker	Used to load images
video_player	Necessary to play the videos of mentors courses
google_maps_flutter	Used to add maps
geocoding	Used to convert addresses into coordinates to place on a map and viceversa
flex_color_scheme	Used for both light and dark themes
cupertino	Used to implement the BottomNavigationBarApp
flutter_test	Used to perform widget tests
integration_test	Used to perform integration tests

3 | User Interface (UI)

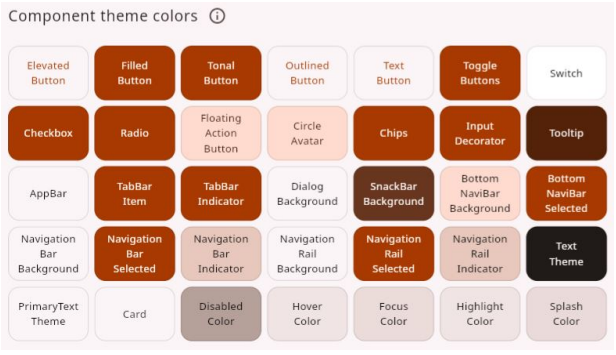
HobbyBuddy's user interface is designed to be intuitive and easy to use, with a modern and clean design language that emphasizes simplicity and clarity. The application's UI is designed to be consistent across different platforms, with a responsive layout that adapts to different screen sizes and resolutions.

This section is used to show the design of the main screens of the user application and the navigation flow. The user can navigate between different screens using a bottom navigation bar, visible on all screens (except for the login and sign up pages) that consists of icons representing the main features of the application:

- **Home Page** (presented to the user after every login)
- **Maps**
- **Friends**
- **Profile Page**

3.1. Visual Design

HobbyBuddy's visual design is minimal, yet fun and captivating. The application is based on a color scheme that is uniform throughout every screen, with the possibility of choosing between a light and a dark theme, allowing great ease of use and consistency. The colors and basic shapes are Material Design 3 compliant thanks to the use of the Amber-Blue theme from the Flex_Color_Scheme package.



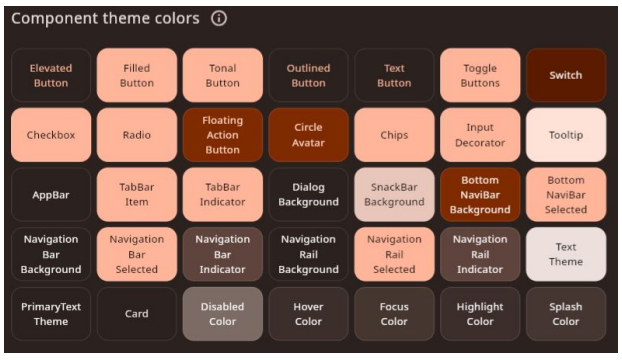


Figure 3.1: Light theme

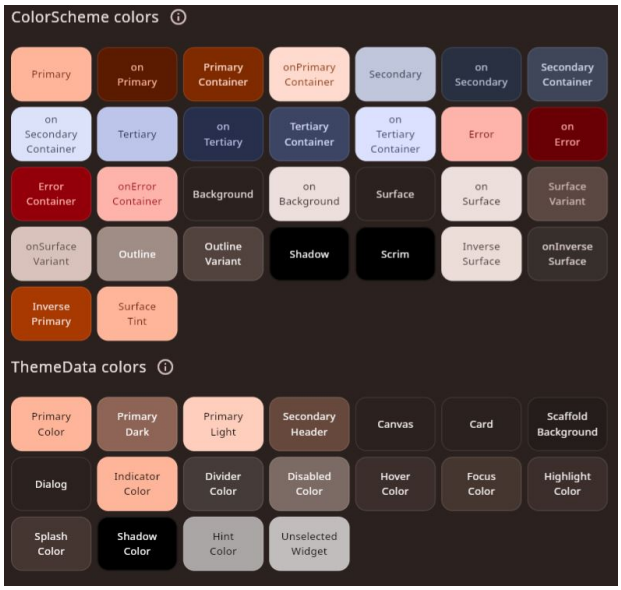


Figure 3.2: Dark theme

3.2. Screens Layout

3.2.1. Login and registration

The first screen displayed when the application is opened for the first time or after a logout is the login page. A button is used to switch from the login to the registration form and vice-versa.

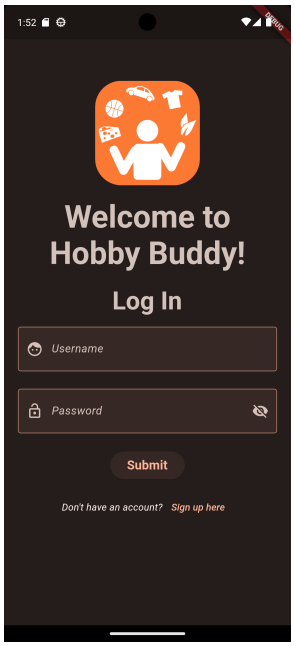


Figure 3.3: Log in

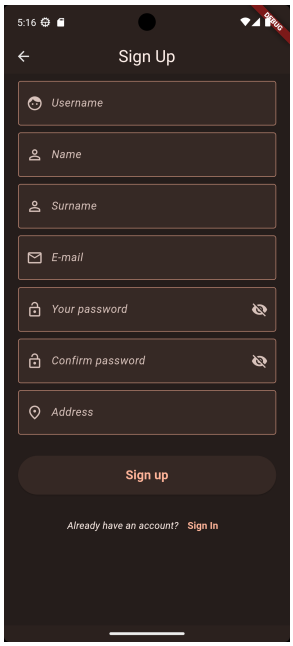


Figure 3.4: Sign up

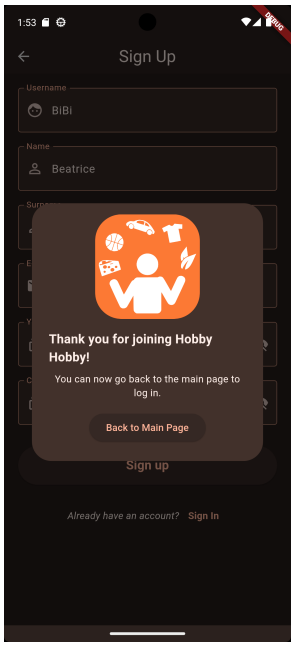


Figure 3.5: Sign up success

3.2.2. Home page and hobby page

HobbyBuddy's **home page** is the first screen displayed to a logged user and contains the list of all hobbies, with an icon indicating whether or not they are part of the user's favourites. A text field allows the user to search for a specific hobby and the icon button right below sorts hobbies alphabetically. All hobbies containers in the grid can be tapped to navigate to the hobby home page. The home page can be re-accessed at any time by tapping the first icon in the bottom navigation bar.
In the **hobby page** the user can like or unlike both the hobby and the mentors by tapping on the respective heart icon and navigate to the mentors' pages by tapping on their list tile.

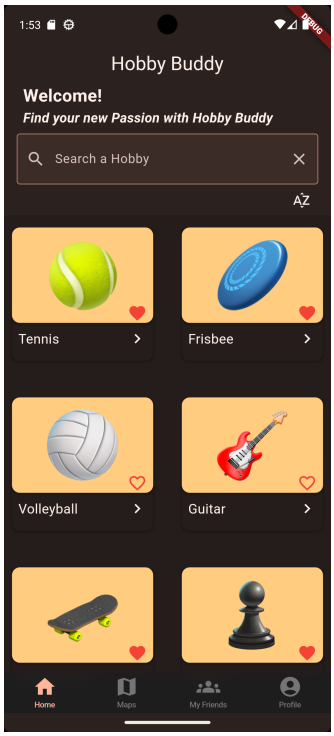


Figure 3.6: Home page

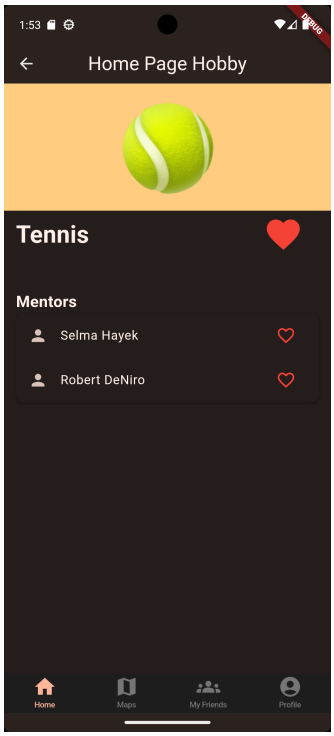


Figure 3.7: Hobby page

3.2.3. Mentor page, Course Page and Maps screen

The **mentors page** contains all basic profile information like profile and background pictures, name, surname and hobby they focus on (mentors can be liked/unliked from this page as well). The page also contains a list of upcoming in-person classes hosted by the mentors and a number of different courses created by the mentor to learn new skills: both classes and courses are color-coded based on difficulty level. The user can tap on a course and open the **course page**: the user can then read the instructions, check the pictures and and watch a video tutorial.
The **maps screen** is the second screen accessible directly from its icon on the bottom navigation bar. Based on the hobbies the user likes, the map shows the locations were classes on those hobbies

are held, by showing a marker with the corresponding icon. By clicking on a marker, the mentor holding classes in that location is shown in a window and the user can tap on it to be redirected to the mentor's page.

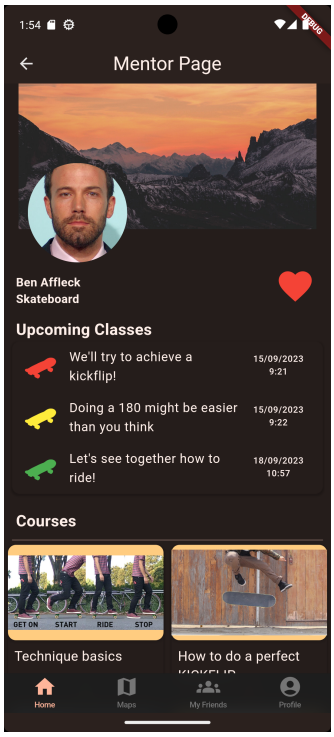


Figure 3.8: Mentor page

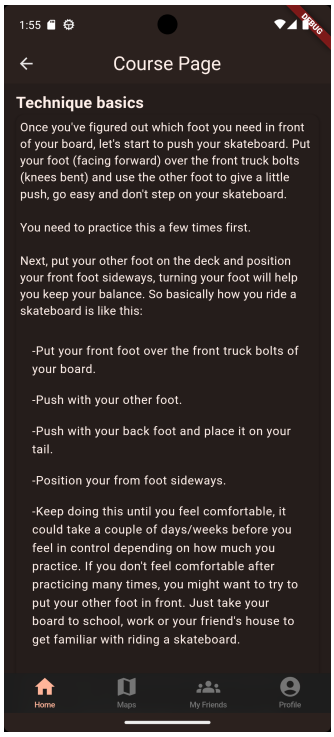


Figure 3.9: Course page pt.1

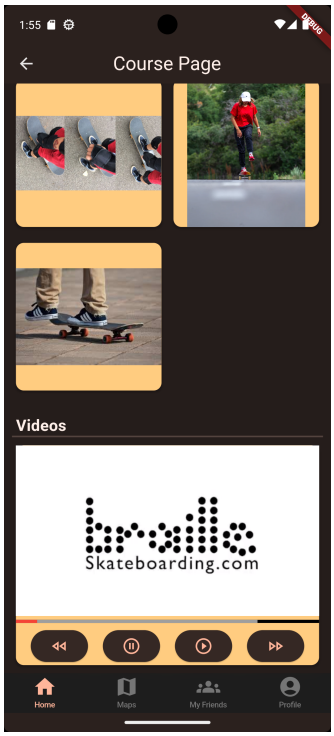


Figure 3.10: Course page pt.2

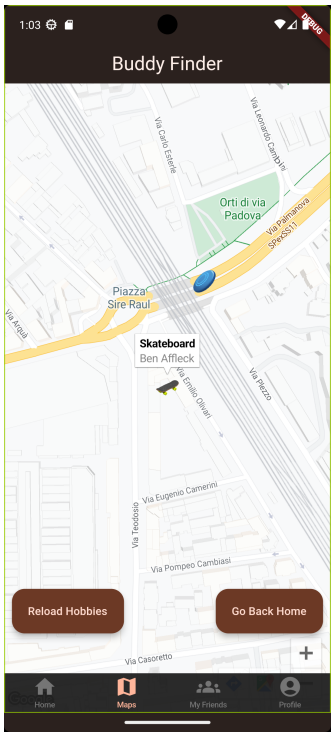


Figure 3.11: Maps

3.2.4. My Friends and Explore

The **Friends** screen can be accessed from the third icon of the bottom navigation bar. It consists of an appBar where the iconButton notifies the user with a red dot when there is a pending friendship request to check: tapping on the icon opens a dialog that displays the list of people waiting for an answer and the corresponding "Accept" and "Decline" buttons.
The rest of the page is divided in two different tabs, both presenting a search bar and an icon to order users alphabetically:

- **My friends** displays the friends of the user. The user can remove a friend from their list of friends (also removing themselves from the other user's list).
- **Explore** displays all other people registered to the app. The user can send a request to another user, which changes the add icon to a pending one, or revoke their request.

By tapping on anyone in the two lists, the user is redirected to that person's profile page (identical to the user's profile page in content structure except for the settings and "+ milestone" buttons) where milestones and information on favourite hobbies and mentors are available.

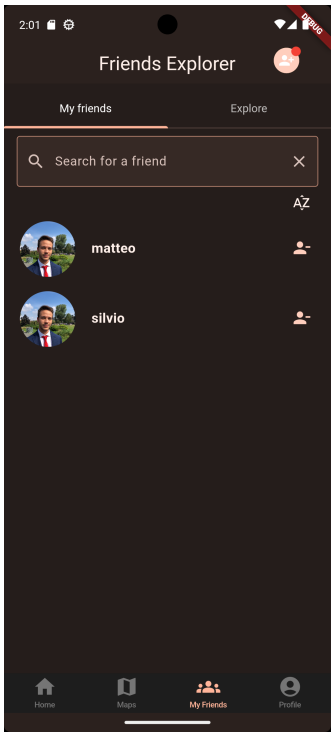


Figure 3.12: My friends

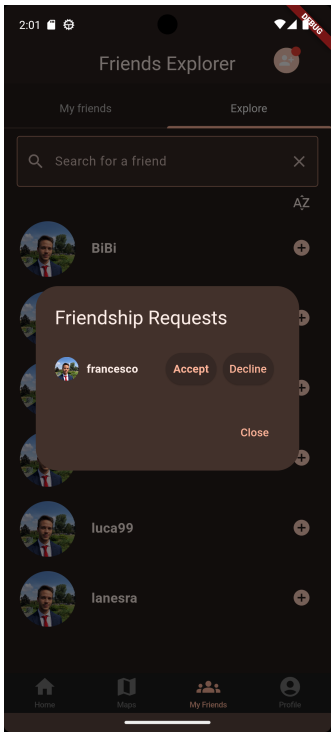


Figure 3.13: Pending requests

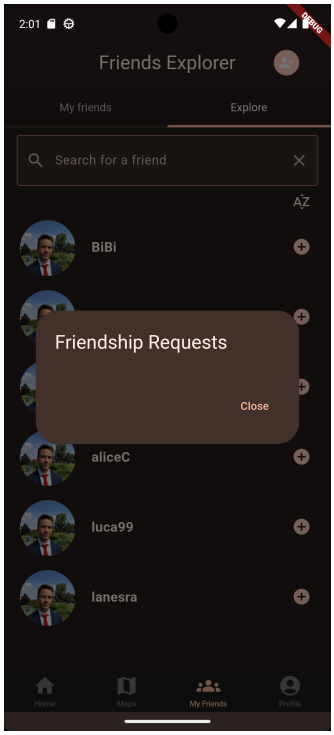


Figure 3.14: No more requests

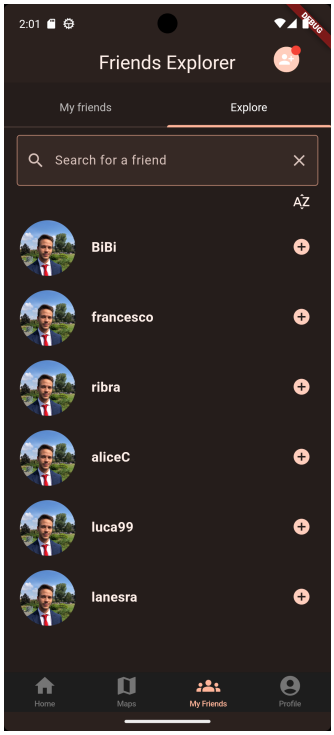


Figure 3.15: Find new friends

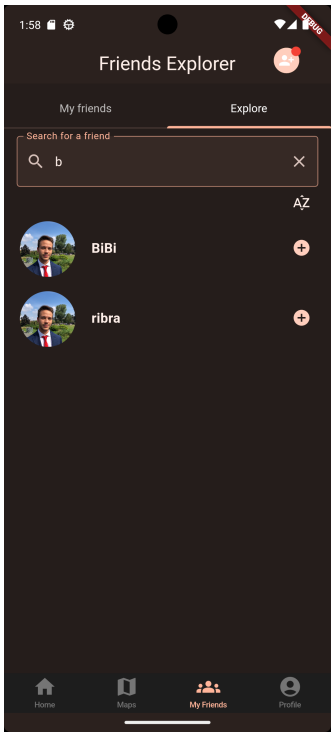


Figure 3.16: Search user

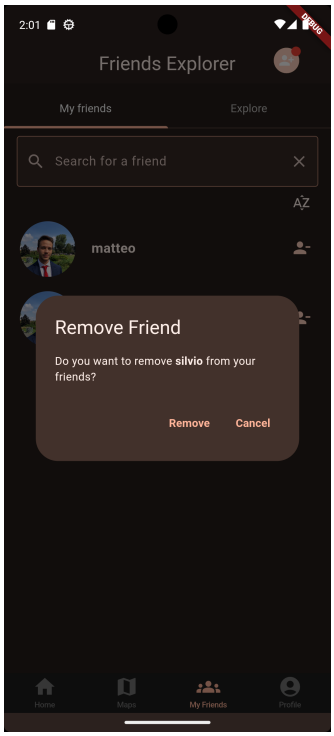


Figure 3.17: Remove friend

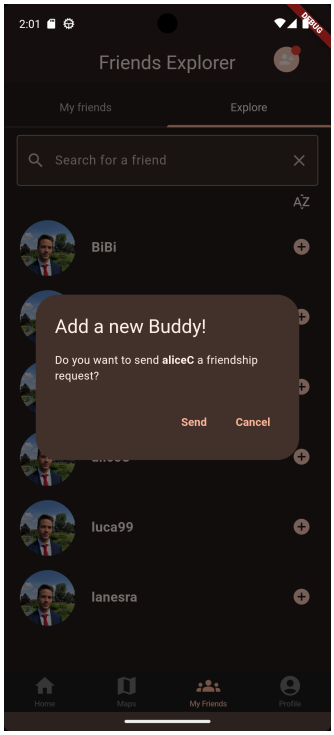


Figure 3.18: Send Request

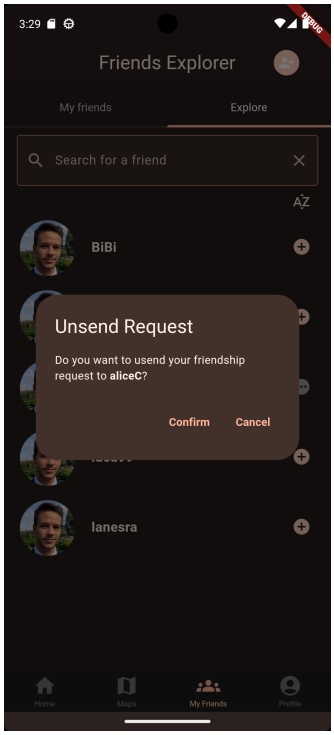


Figure 3.19: Unsend Request

3.2.5. Profile page

The **Profile page** is the last page of the bottom navigation bar and it displays all the user's pictures, name, surname and address.
Two horizontally scrollable lists show all the user's favourite hobbies and mentors, with the possibility of being redirected to their relative pages by tapping on the corresponding icon.
The rest of the page can be scrolled vertically to better visualize all of the user's personal milestones: each milestone has the timestamp of the moment of upload, a caption and a picture. Milestones are visualized from the most recent to the oldest. Finally, there is the possibility to add a new milestone by tapping on the button at the beginning of the section: it redirects the user to a new screen where a caption can be entered and a picture can be chosen from the gallery, in order to upload a new milestone.

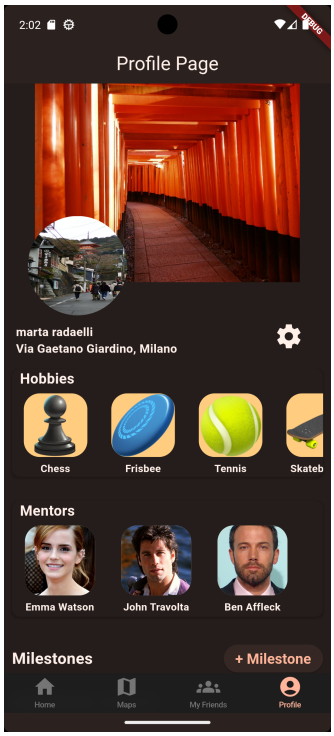


Figure 3.20: Profile page

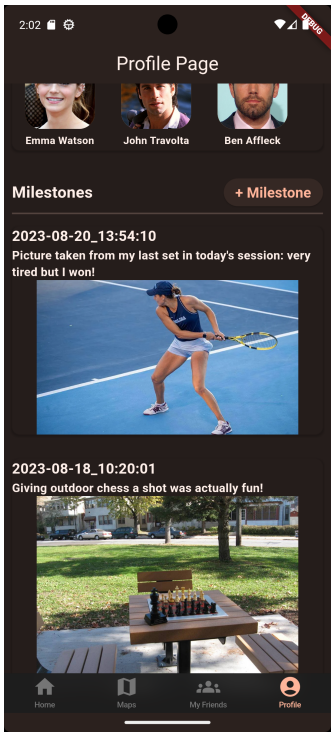


Figure 3.21: Profile page

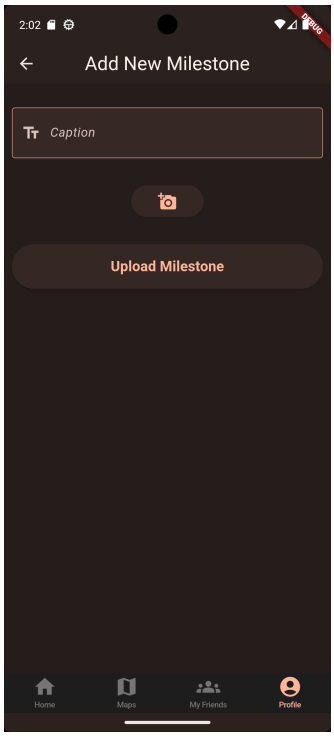


Figure 3.22: Upload milestone

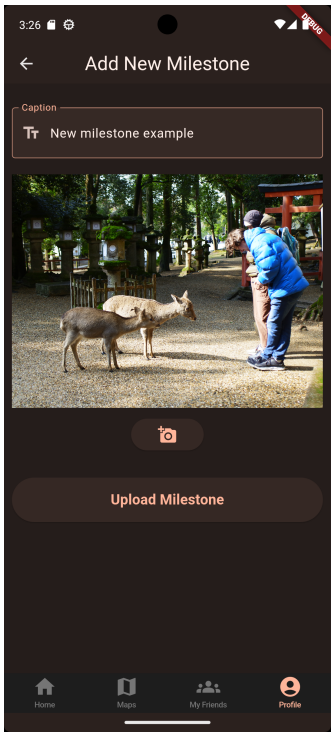


Figure 3.23: Review upload

3.2.6. Settings

The **Settings** screen can be accessed from the icon in the user profile page. The screen allows the user to change the setting of the theme from light to dark (and vice-versa) by tapping the switch. The user can change screen and modify its password to a new one or tap on edit profile and navigate to a new screen that allows to change name and surname as well as upload new pictures from the device gallery. The user gets a preview of how the new images look and can decide whether to save the new settings or go back to the old picture by tapping the "x" icon next to the new images.

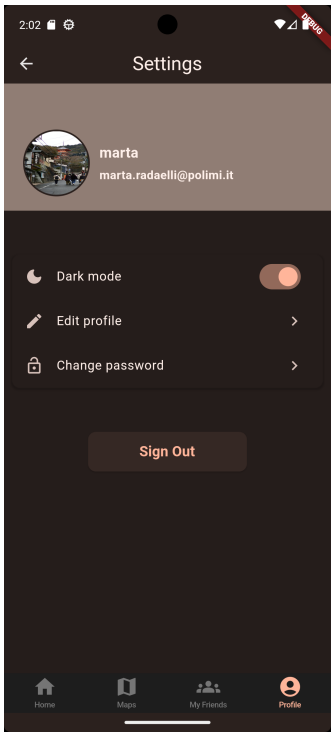


Figure 3.24: Settings

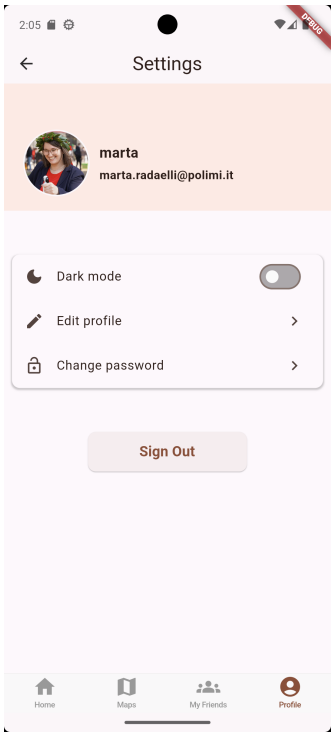


Figure 3.25: Light theme

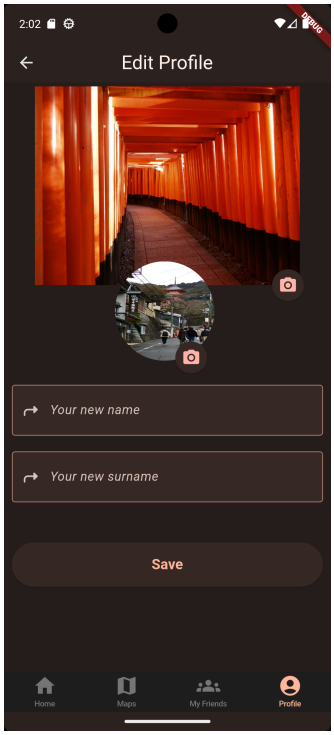


Figure 3.26: Edit screen

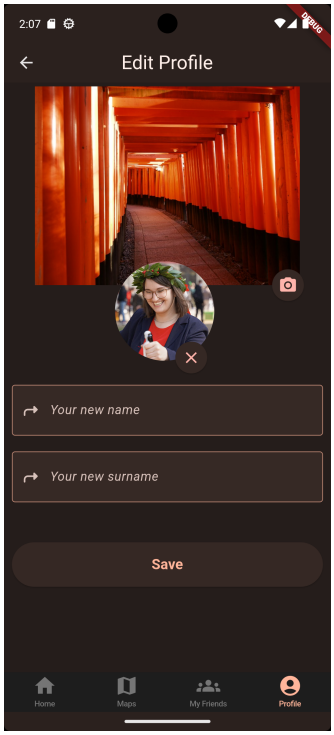
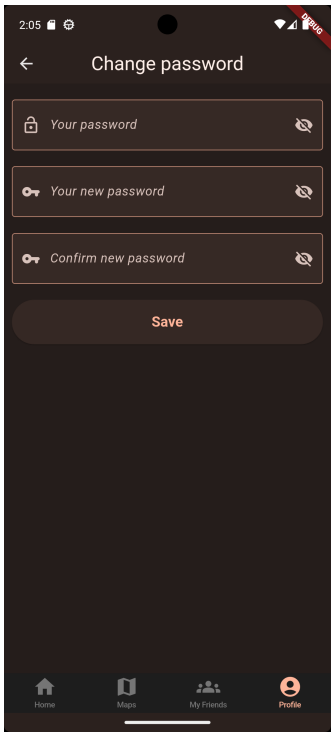


Figure 3.27: Changed image



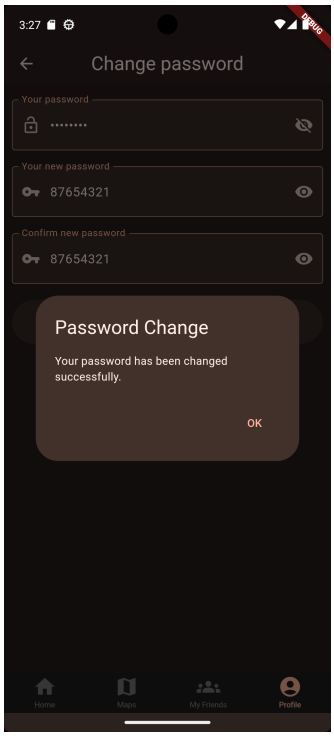
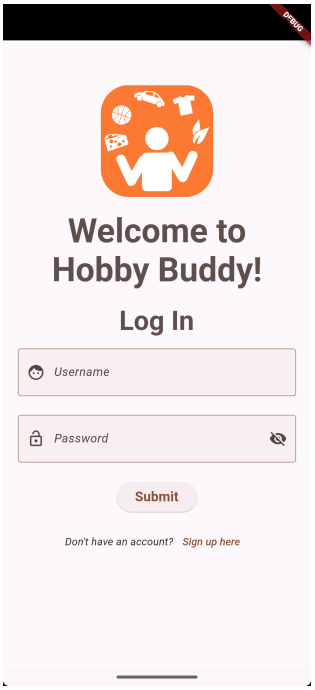
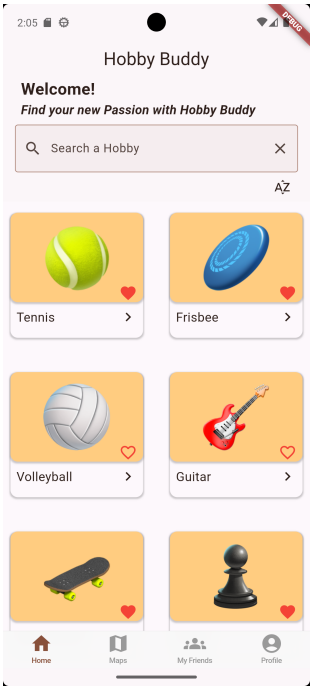


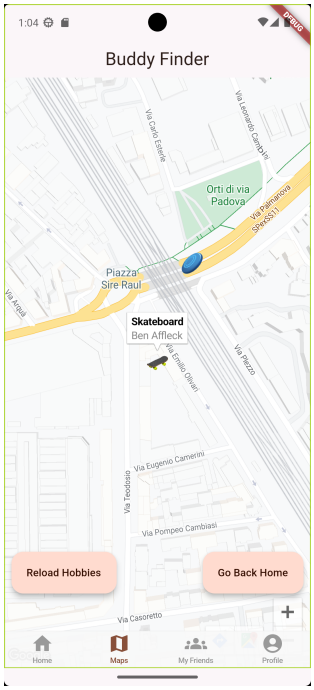
Figure 3.28: Change password Figure 3.29: Success dialog

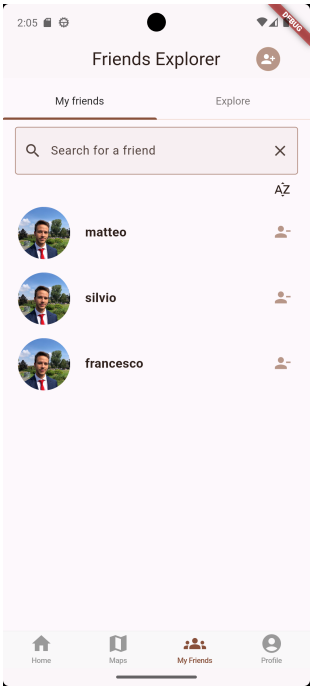
3.2.7. Light Mode

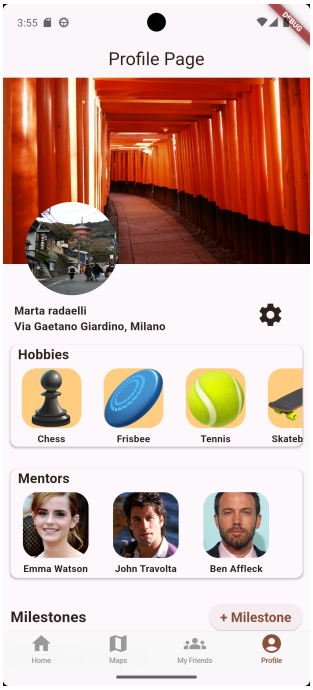
A few of the screens just seen are presented in this this section in light mode in order to get an idea of how the contrasting theme looks like in our application.


Welcome to Hobby Buddy!
Log In
Username
Password
Submit
Don't have an account? Sign up here


Hobby Buddy
Welcome!
Find your new Passion with Hobby Buddy
Search a Hobby
Tennis
Frisbee
Volleyball
Guitar
Home
Maps
My Friends
Profile

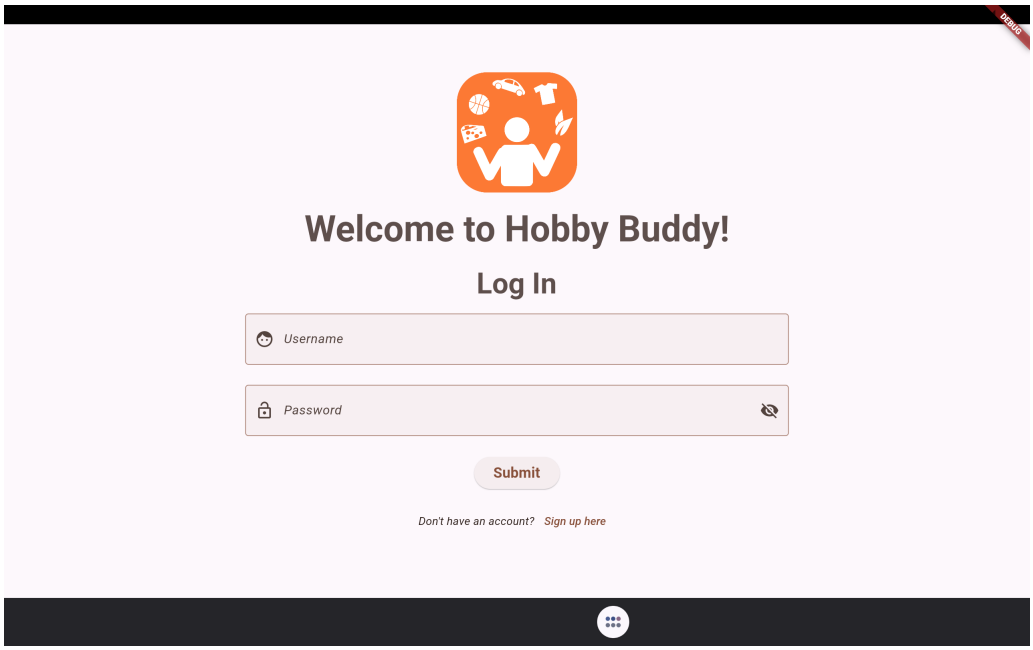

Buddy Finder
Skateboard
Ben Affleck
Reload Hobbies
Go Back Home

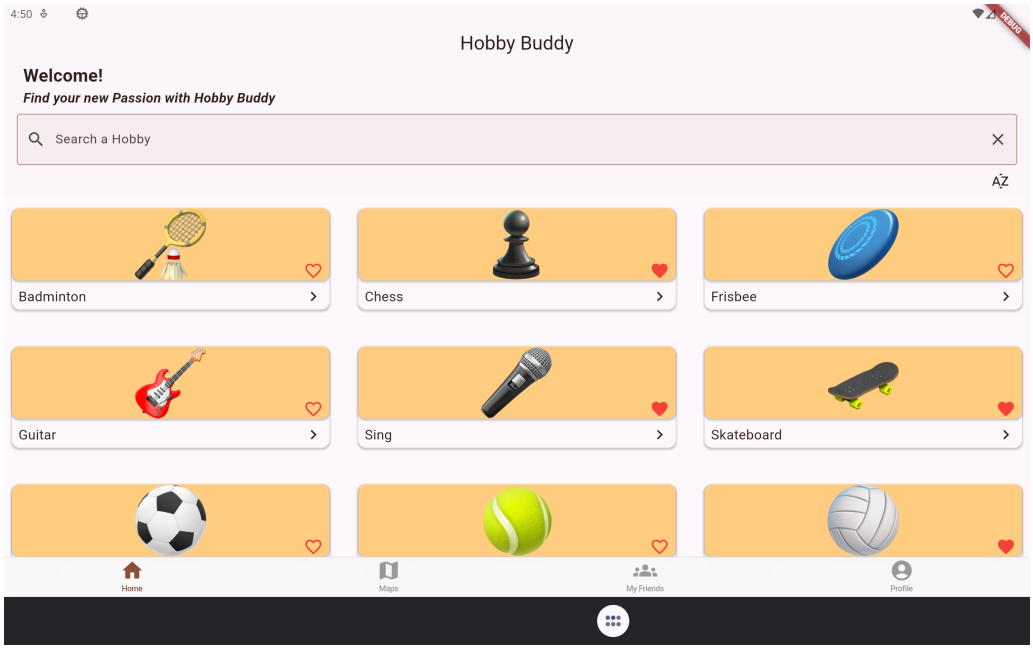

Friends Explorer
My friends
Explore
Search for a friend
matteo
silvio
francesco
Home
Maps
My Friends
Profile

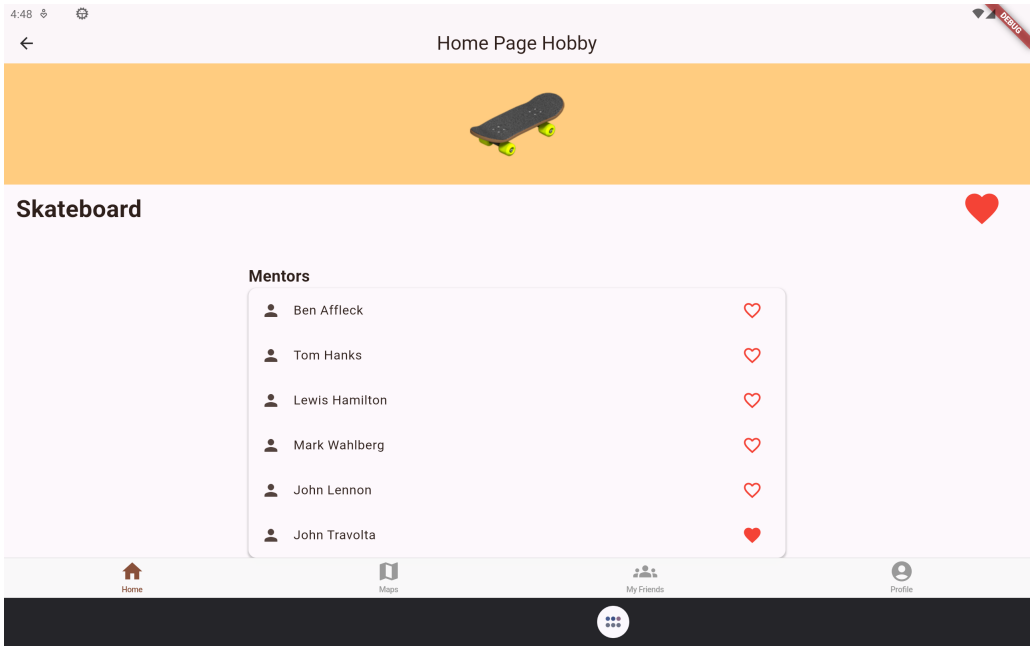

Profile Page
Marta radaelli
Via Gaetano Giardino, Milano
Hobbies
Chess
Frisbee
Tennis
Mentors
Emma Watson
John Travolta
Ben Affleck
Milestones
+ Milestone
Home
Maps
My Friends
Profile

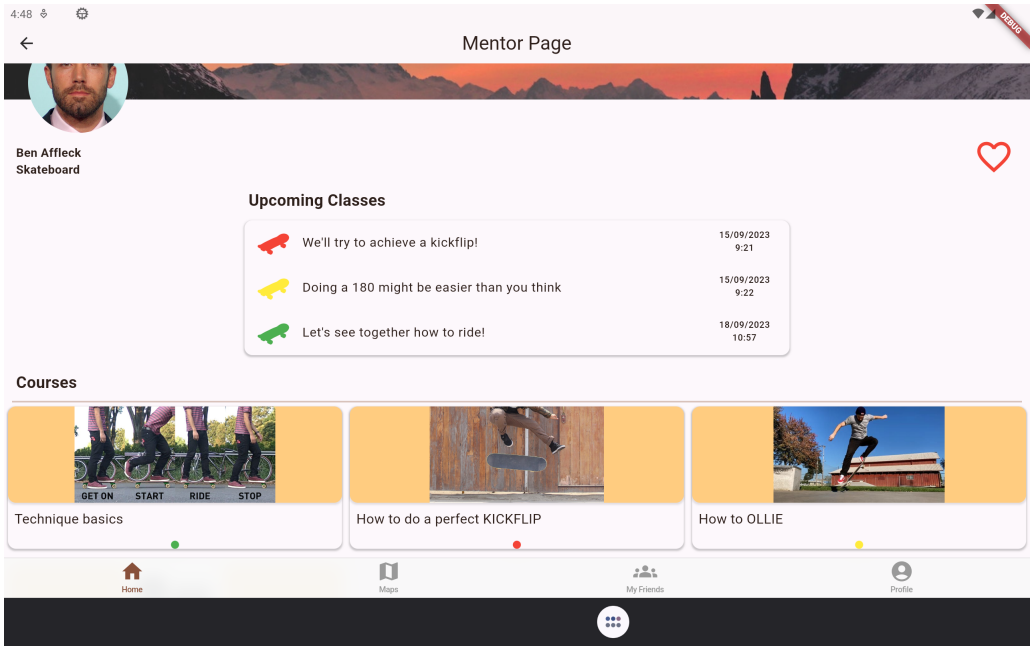
3.2.8. Tablet UI

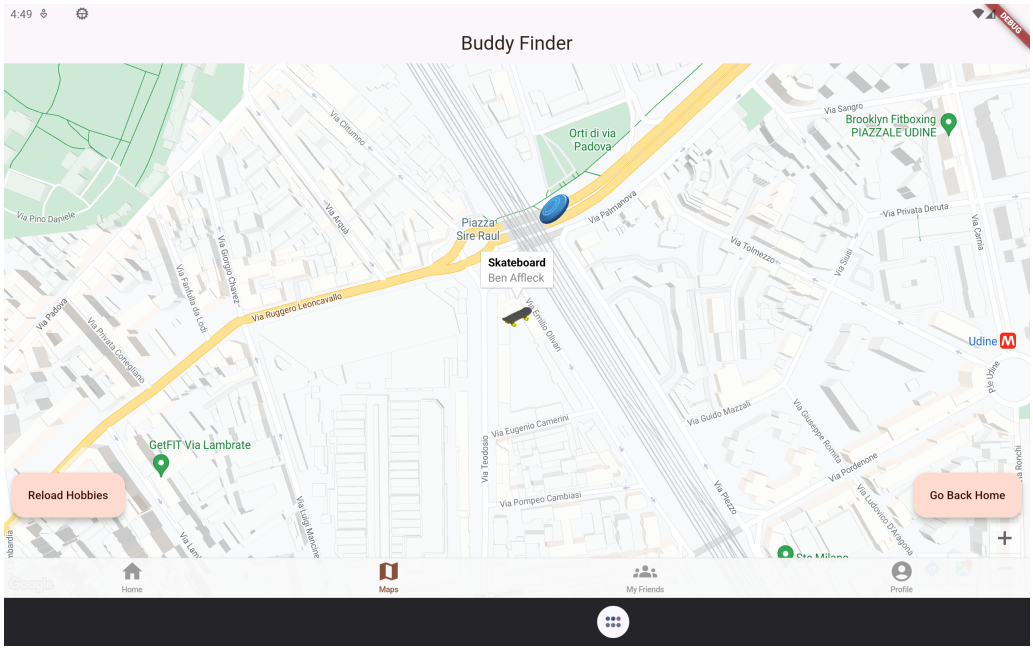
HobbyBuddy is optimized to work well on any kind of device and its User Interface changes according to the screen size, improving the usability and the look-and-feel of the application without, at the same time, altering the core structure of the application. In this section are shown some of the most relevant changes with regards to the smartphone design.

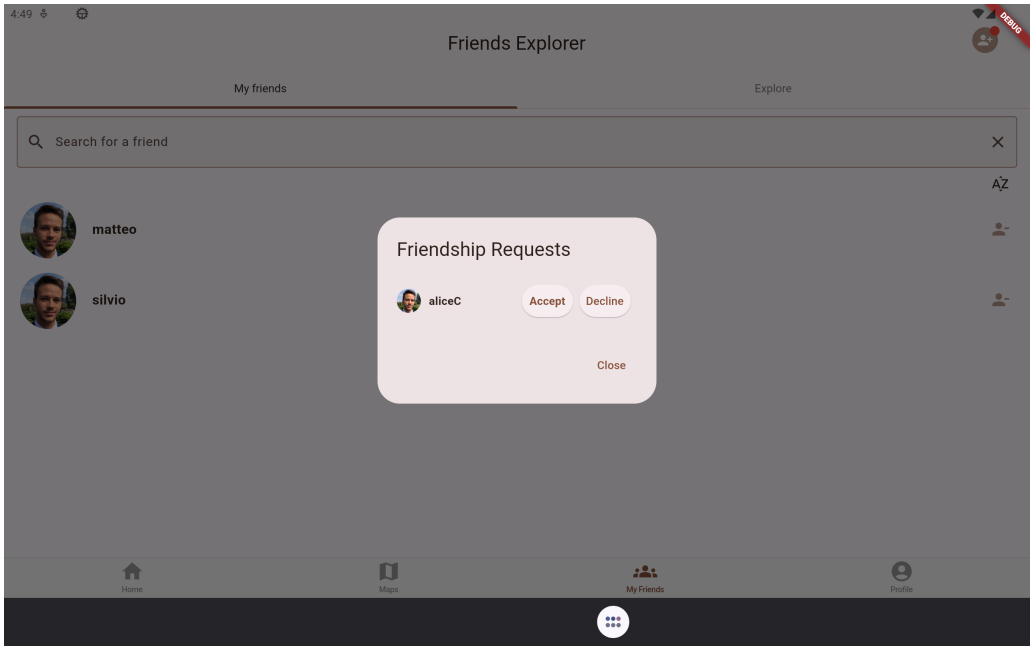


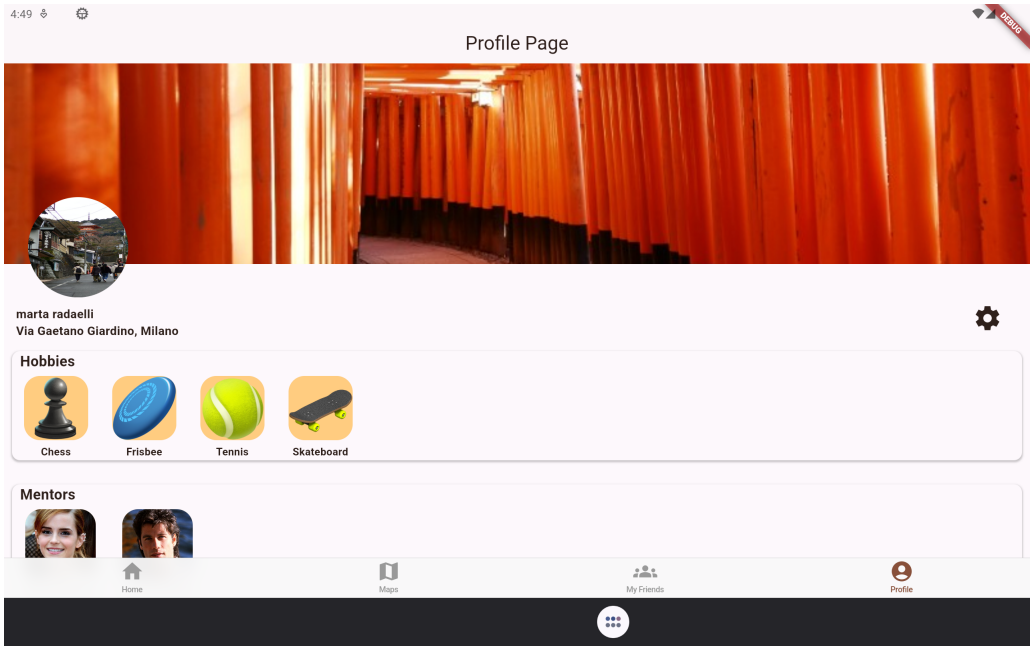


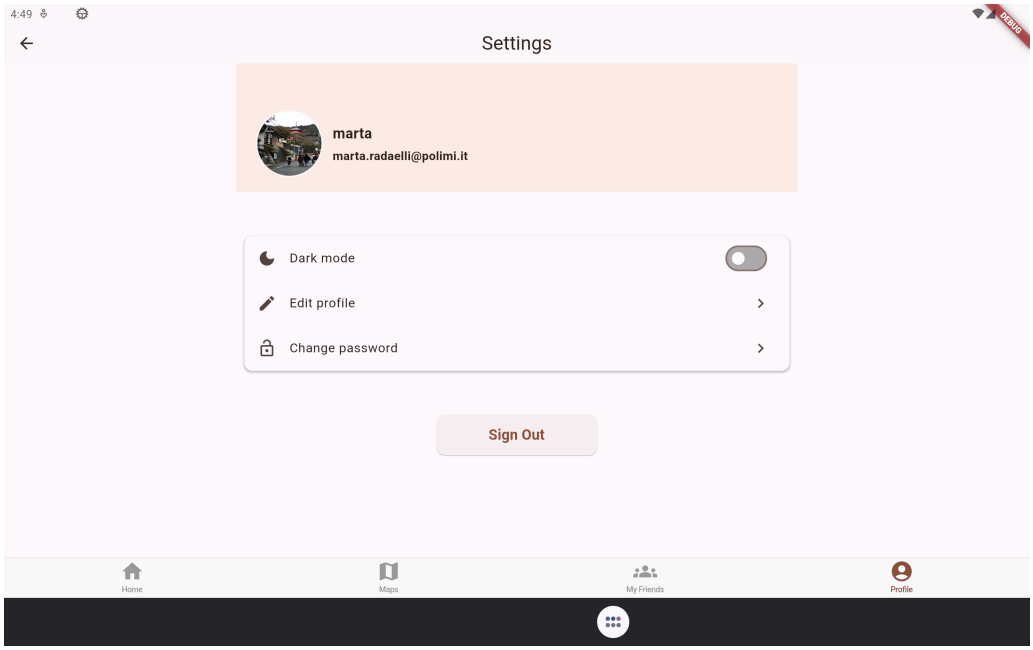

4:48
Home Page Hobby
Skateboard
Mentors
Ben Affleck
Tom Hanks
Lewis Hamilton
Mark Wahlberg
John Lennon
John Travolta
Home
Maps
My Friends
Profile


4:48
Mentor Page
Ben Affleck
Skateboard
Upcoming Classes
We'll try to achieve a kickflip!
15/09/2023
9:21
Doing a 180 might be easier than you think
15/09/2023
9:22
Let's see together how to ride!
18/09/2023
10:57
Courses
GET ON
START
RIDE
STOP
Technique basics
How to do a perfect KICKFLIP
How to OLLIE
Home
Maps
My Friends
Profile


4:49
Buddy Finder
Skateboard
Ben Affleck
Piazza Sire Raul
Orti di via Padova
Brooklyn Fitboxing PIAZZALE UDINE
GetFIT Via Lambrate
Udine
Via Ruggero Leoncavallo
Reload Hobbies
Go Back Home
Home
Maps
My Friends
Profile


4:49
Friends Explorer
My friends
Explore
Search for a friend
matteo
silvio
Friendship Requests
aliceC
Accept
Decline
Close
Home
Maps
My Friends
Profile


4:49
Profile Page
marta radaelli
Via Gaetano Giardino, Milano
Hobbies
Chess
Frisbee
Tennis
Skateboard
Mentors
Home
Maps
My Friends
Profile


4:49
Settings
marta
marta.radaelli@polimi.it
Dark mode
Edit profile
Change password
Sign Out
Home
Maps
My Friends
Profile

4 | Testing Campaign

The testing plan for HobbyBuddy includes a set of extensive test cases executed both throughout and at the end of the app's development. The tests developed for the application are performed in 2 different environments according to the kind of test performed:

- **Mock Environment**: all the APIs are replaced by a mock version in order to separate the behavior of the external service from the tested implemented function and to overcome the many limitation of the flutter test environment, where no external services are working.
  Thanks to the architecture of the application, this environment is implemented simply by changing the implementation of our service classes to mock versions with static data available.
  The mocked APIs were:
  - **Firebase firestore** with the use of package **fake_cloud_firestore**
  - **Firebase storage** thanks to the library **firebase_storage_mocks**
  - **Firebase authentication** by using library **firebase_auth_mocks**
  - **Google maps for flutter** thanks to the mocked **fake_google_maps_platform** code found on the GitHub repository of the package.
  - **Image picker** and **geocoding** through custom mock functions created with package **mockito**
- **Deploy Environment**: here tests are supposed to run on the real device. This costly and slow conditions are necessary for testing the integration of the screens in as well as the integration between our app with the external services.

4.1. Unit Testing

This part of the campaign aims at verifying the correct behavior of the basic interface and model classes by testing them in isolation: Flutter unit tests are supposed to run in the mock environment due to flutter constraints. Our app heavily depends on external services so most of the logic implemented is a trivial call to them with different parameters. Because of these limitations, we were able to perform a limited number of unit tests.

4.2. Widget Testing

Widget tests check if the widget's UI looks and interacts as expected by testing multiple classes and require an environment that provides the appropriate widget lifecycle context. The aim of this type of testing is to ensure that a proper widget is loaded and populated successfully, by feeding it data coming from mock APIs and checking through proper assertions if it's correctly displayed. The Flutter Widget tests implemented run in the mock environment, testing the different screens on different granularity levels and on all possible use cases.

Tested Screen	Short description of the test
Log In	Log in screen renders correctly with button
Log In	Log in screen logs existing user
Log In	Log in screen rejects non existing user
Log In	Log in screen navigates to sign up screen
Sign Up	Sign Up Screen displays form and handles a correct sign-up
Sign Up	Sign Up Screen rejects already existing username
Sign Up	Sign Up Screen handles empty fields and mismatched passwords
Sign Up	Sign Up Screen handles handles going back to log in screen
Home Page	Home page renders correctly and a hobby container can be tapped
Home Page	Hobbies in home page can be searched and sorted alphabetically
Hobby Page	Hobby's homepage renders correctly
Hobby Page	Hobby and mentors can be liked/unliked
Mentor Page	Mentor's homepage renders correctly
Mentor Page	Mentor's homepage like button works and user can navigate to a course
Course Page	Course page renders correctly
Course Page	User can interact correctly with video player
Map Page	Map page renders and behaves correctly on user interaction
My Friends Page	My friends screen renders correctly and allows friends and users to be searched an sorted alphabetically in both tabs
My Friends Page	User can interact with friendship request dialog to accept or decline a request
My Friends Page	User can delete a friend
My Friends Page	User can send friendship request and revoke a pending request
My Friends Page	My friends screen refreshes correctly
My Friends Page	Explore screen refreshes correctly
User Profile	User's homepage renders correctly
User Profile	User can navigate to a favourite hobby
User Profile	User can navigate to a favourite mentor
User Profile	User can add navigate to the "add new milestone screen"

Add New Milestone	Add milestone's page renders correctly
Add New Milestone	Add milestone's page handles the upload of an empty milestone fields
Add New Milestone	Add milestone's page asks user to write a caption if the field is empty
Add New Milestone	Add milestone's page asks user to load an image of the field is empty
Add New Milestone	Add milestone's page uploads new milestone when both the caption and image are present
Settings	Settings screen renders correctly and can tap on sign out
Settings	Dark/light mode switch can be tapped
Settings	Tap to navigate to edit profile screen
Settings	Tap to navigate to change password screen
Edit Profile	Edit Profile renders correctly and saves new name and surname
Edit Profile	Edit Profile handles image changes
Change Password	Change password screen displays form correctly and handles a correct update
Change Password	Change password screen displays form correctly and handles a mismatched new passwords
Change Password	Change password screen displays form correctly and handles passwords shorter than 8 characters

4.2.1. Coverage Analysis

We were able to reach a test coverage of around 90%, also thanks to the flutter coverage library. The untested lines where checked manually as they are, for the most part, fallback methods needed in case of server or connection errors.

LCOV - code coverage report

Current view: top level - lib/screens/lib/screens
Test: lcov.info
Test Date: 2023-09-04 10:03:45

	Coverage	Total	Hit
Lines:	95.1 %	1896	1804
Functions:	-	0	0

Filename	Line Coverage			Function Coverage		
	Rate	Total	Hit	Rate	Total	Hit
add_milestone.dart	97.0 %	66	64	-		
change_password.dart	90.6 %	96	87	-		
courses.dart	96.6 %	179	173	-		
edit_profile.dart	93.2 %	147	137	-		
friends_list.dart	98.4 %	128	126	-		
home_page.dart	90.7 %	129	117	-		
homepage_hobby.dart	92.3 %	78	72	-		
homepage_mentor.dart	96.5 %	202	195	-		
homepage_user.dart	94.0 %	266	250	-		
login.dart	89.1 %	92	82	-		
maps.dart	97.6 %	42	41	-		
my_friends_list.dart	96.4 %	110	106	-		
search_friend_list.dart	97.2 %	144	140	-		
settings.dart	97.2 %	71	69	-		
sign_up.dart	99.3 %	146	145	-		

Generated by: LCOV version 2.0-1

LCOV - code coverage report

		Coverage	Total	Hit
Current view:	top level - lib/services/lib/services			
Test:	lcov.info	Lines:	82.8 %	349	289
Test Date:	2023-09-04 10:03:45	Functions:	-	0	0

Filename	Line Coverage			Function Coverage		
	Rate	Total	Hit	Rate	Total	Hit
firebase_auth.dart	66.7 %	3	2	-		
firebase_firestore.dart	80.9 %	303	245	-		
firebase_storage.dart	80.0 %	5	4	-		
light_dark_manager.dart	100.0 %	5	5	-		
preferences.dart	100.0 %	33	33	-		

Generated by: LCOV version 2.0-1

LCOV - code coverage report

		Coverage	Total	Hit
Current view:	top level - lib/widgets/lib/widgets			
Test:	lcov.info	Lines:	100.0 %	62	62
Test Date:	2023-09-04 10:03:45	Functions:	-	0	0

Filename	Line Coverage			Function Coverage		
	Rate	Total	Hit	Rate	Total	Hit
app_bar.dart	100.0 %	9	9	-		
button.dart	100.0 %	10	10	-		
button_icon.dart	100.0 %	9	9	-		
container_shadow.dart	100.0 %	13	13	-		
responsive_wrapper.dart	100.0 %	10	10	-		
screen_transition.dart	100.0 %	11	11	-		

Generated by: LCOV version 2.0-1

4.3. Integration Testings

Integration test are used to simulate a user interacting with the app checking how individual pieces work together as a whole. This phase is an extension of widget tests, completing the testing on the app: in this case, we will focus on how different widgets interact with each other to verify that they all work together as expected.
These kinds of tests are done in the deployment environment so that we can manage to test the stability of the system with the external APIs and check whether the app has good performance.

Because of the nature of integration testing, the focus was given to the steps needed to perform the main actions of the application starting from the initial page. The resulting files and tests are:

- SIGN UP
  - Sign up new user, login and go to the home page
  - Fail to sign up with already existing username
- LOG IN
  - Log in with non existent account fails
  - Log in successfully with a correct account, go to home page and then navigate to log out
- NAVIGATIONS
  - From the home page, navigate to a hobby, like or unlike it, then tap on a mentor, like or unlike it as well, and scroll to tap on an available tutorial and read it. Navigates back to

home page via bottomAppBar.

- Navigate to the friends page via bottomAppBar, change the visualized tab, accept a friendship request, refresh the screen to check if you received others in the meantime and send a friendship request to a user. Then change back to the other tab and remove a friend from your list and then visit the user page of a friend.
- Navigate to the profile page via bottomAppBar, open a hobby page, go back to the profile page, scroll down to see milestones and tap "+ milestone" to add a new one.
- Navigate to the user page, then to the settings in order to tap the switch for the dark mode theme, edit the user name and change the password. Then navigate back to the user page.

4.4. User Testing

This phase of the testing is performed to check how people that have no bias from the app implementation interface with the interface design.

These tests were performed taking a very restricted number people foreign to the application and let them use the app in 2 ways:

- perform actions guided by the Integration Test list -> useful to check if the most relevant functions of the application are easily accessible to new new users.
- free navigation in the app -> used to check if the design of the app is able to guide the user to perform the action he wants to do.

Due to the involvement of external people, this was by far one of the most complicated tests to perform, but it also gave us great feedback on how our design could benefit from some changes that would improve the app experience and accessibility.

5 | Future Developments

HobbyBuddy allowed us to develop several features in different ranges of interest in the world of mobile application design and development. The source code of the first release of HobbyBuddy covers most of the features we had in mind, but due to time constraint we postponed the developments of less relevant features.
Thanks to the modular architecture of the application, all the future developments will require very few, if not any, alterations of the source code written since now. A list of possible future improvement is:

- Substitute the back-end of the application with a custom one not dependent on Firebase, for more customizable backend operations.
- Implement administrator services for mentors: one of the design choices of the app is that all data connected to a mentor is managed directly by a dedicated team on the back-end. We would like to develop a mentor-dedicated version of the app to improve all users, mentors and mentee, involvement with the app with the possibility of adding features like online live lessons with mentors or a Q&A section where a user can ask a mentor any kind of question and get tips.
- Allowing mentors to choose a location for each of their in-person classes, instead of having only one location linked to their profile.
- Start collaborations with people that are well known in a specific discipline to increase interest in the app: a project like this would require great monetary and time investment that were outside the scope of the "design and implementation of mobile application" course.
- Adding support for in-app messaging and chat services.
- Leveraging predictive analytics and recommender systems to predict potential hobbies and courses that the user may have an interest in.
- Add the option of in-app purchase for advanced and special courses or merchandise.
- Improve the support for different screen sizes and orientation, especially for tablets.